

# SALTAMAR INNOVATIONS

IC10 Rec'd PCT/PTO 25 FEB 2002

30 Fern lane  
South Portland, ME 04106  
USA  
Phone: +1(207) 799-9733  
Fax: +1(207) 228-3694  
E-mail: Shalom@Saltamar.com  
HTTP://www.saltamar.com

*In the United States Patent and Trademark Office*

In re application of: <b>Pekka Marjola</b>	
For: <b>Method and Arrangement for Sending a Video Presentation</b>	
Serial No.~Conf: <b>10/041,021~2621</b>	Group:
Filed on: <b>Jan. 7, 2002</b>	Examiner:
Correspondence Date <b>Feb 24, 2002</b>	Docket: <b>0201US-Oplayo</b>

## Priority Document Filing in accordance with 37 CFR 1.55

Commissioner of Patents  
Washington DC, 20231

Sir,

Applicant hereby requests to enter into record the enclosed certified copy of Finnish patent application No. 20012558, filed Dec. 21, 2001. Applicant claims the benefit of priority to this application in accordance with 35 USC 119.

Respectfully submitted

Shalom Wertsberger

30 Fern Lane  
South Portland, ME 04106  
Phone: (207) 799-9733  
Fax: (207) 799-3698

Agent for Applicant  
Reg. Num. 43,359

RECEIVED

APR 11 2002

Technology Center 2100

Certificate under 37 CFR 1.10 of Mailing by "Express Mail"

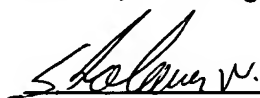
EJ 676543581US

"Express Mail" label number

February 25, 2002

Date of Deposit

I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231



Signature of person mailing correspondence

Shalom Wertsberger Reg. No. 43,359

Typed or printed name of person mailing correspondence

RECEIVED

APR 11 2002

Technology Center 2100

Docket: 0201US-Oplayo

Application #: 10/041,021 ~ 2621

Inventor: Pekka MARJOLA

Filed on: January 07, 2002

Title: METHOD AND ARRANGEMENT FOR SENDING A VIDEO PRESENTATION

This correspondence contains:

Certified copy of priority document, Finnish patent application No. 20012558, filed Dec. 21, 2001

☐ Check No. \_\_\_\_\_ or ☐ credit card charge authorization for \_\_\_\_\_

PATENTTI- JA REKISTERIHALLITUS  
NATIONAL BOARD OF PATENTS AND REGISTRATION

Helsinki 11.1.2002

ETUOIKEUSTODISTUS  
PRIORITY DOCUMENT



Hakija  
Applicant

Oplayo Oy  
Helsinki

Patenttihakemus nro  
Patent application no

20012558

Tekemispäivä  
Filing date

21.12.2001

Kansainvälinen luokka  
International class

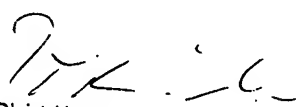
H04L

Keksinnön nimitys  
Title of invention

"Method and arrangement for sending a video presentation"  
(Menetelmä ja järjestely videoesityksen lähettämiseksi)

Täten todistetaan, että oheiset asiakirjat ovat tarkkoja jäljennöksiä patentti- ja rekisterihallitukselle alkuaan annetuista selityksestä, patenttivaatimuksista, tiivistelmästä ja piirustuksista.

This is to certify that the annexed documents are true copies of the description, claims, abstract and drawings originally filed with the Finnish Patent Office.

  
Pirjo Kaitu  
Tutkimussihteeri

CERTIFIED COPY  
PRIORITY DOCUMENT

Maksu 50 €  
Fee 50 EUR

Maksu perustuu kaupp- ja teollisuusministeriön antamaan asetukseen 1027/2001 Patentti- ja rekisterihallituksen maksullisista suoritteista muutoksineen.

The fee is based on the Decree with amendments of the Ministry of Trade and Industry No. 1027/2001 concerning the chargeable services of the National Board of Patents and Registration of Finland.

Osoite: Arkadiankatu 6 A

Puhelin:

09 6939 500

Telefax:

09 6939 5328

## Method and Arrangement for Sending a Video Presentation

### Field of the Invention

5 This invention relates to sending video and voice or other continuous data over a network, such as the Internet, to subscribers' terminals. Especially the invention concerns sending live video show from a video producer to subscribers.

### Background of the Invention

10 There exist several solutions for sending real-time, i.e. live, video from a video producer to a customer's terminal. The RTP (Real-time Transport Protocol) and RTSP (Real Time Streaming Protocol) are usually the most common solutions used but other solutions exist as well.

15 RTP (Real-time Transport Protocol) provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of- service for real-time services. Sequence numbers included in RTP  
20 allow the receiver to reconstruct the sender's packet sequence. The sequence numbers might also be used to determine the proper location of a packet, for example in video decoding, without necessarily decoding packets in sequence.

25 The RTP data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTCP is based on the periodic transmission of control packets to all participants in the session, using the same distribution mechanism as the data packets. The underlying protocol must provide multiplexing of the data and control packets, for example using separate port numbers with UDP.  
30 RTCP provides feedback on the quality of the data distribution and carries a persistent transport-level identifier for an RTP.

Both RTP and RTCP are designed to be independent of the underlying transport and network layers.

35 The Real Time Streaming Protocol (RTSP) is an application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery

of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP

RTSP establishes and controls either a single or several time-synchronized streams of continuous media (audio and video i.e. data where there is a timing relationship between a sender and receiver). It does not typically deliver the continuous streams itself, although interleaving of the continuous media stream with the control stream is possible. In other words, RTSP acts as a remote control for multimedia servers. The set of streams to be controlled is defined by a presentation description. The stream means a single media instance, such as an audio stream or a video stream. When using RTP, a stream consists of all RTP and RTCP packets created by a sender within an RTP session.

An RTSP session is not tied to a transport-level connection such as a TCP connection. During an RTSP session, an RTSP client may open and close many reliable transport connections to the server to issue RTSP requests. Alternatively, it may use a connectionless transport protocol such as UDP. The streams controlled by RTSP may use RTP, but the operation of RTSP does not depend on the transport mechanism used to carry continuous media.

Since not all media servers have the same functionality, media servers by necessity will support different sets of requests. For example, a server is only capable of playback, or a server is not capable of seeking absolute positioning, if it is to support live events only, or furthermore some servers do not support setting stream parameters and thus not support GET parameter and SET parameter.

For compensating the lack of supporting various requests, RTSP can be extended in some ways. For example, existing methods can be extended with new parameters, as long as these parameters can safely be ignored by the recipient. Another way is to add new methods. If the recipient of the message does not understand the request, it responds with an error code and the sender should not attempt to use this method again. A client may also use the OPTIONS method to inquire about methods supported by the server. The server SHOULD list the methods it supports using the Public

response header. Further, a new version of the protocol can be defined, allowing almost all aspects to change.

As can be noted, the known solutions for delivering live media streams are relatively complex. Furthermore, they are usually not supported  
5 in web servers in the Internet since they need special software. Many service providers have an interest to invest in different, expensive, and parallel media delivery systems. The goal of the invention is to alleviate the problems of the known solutions. This is achieved in a way described in the claims.

#### 10                   **Summary of the Invention**

The idea of the invention is to use a standard web server (supporting an HTTP protocol), which is adapted to send consecutive video files to subscribers' terminals directly, or via a proxy. The server receives the encoded video data arranged into consecutive packets from a video  
15 provider's encoding element, called broadcaster. The sending from the broadcaster to the web server is preferably made using FTP (File Transfer Protocol) protocol. The FTP transport contains both the video data and index data. The index data identifies the current file that is being sent to the subscriber's terminal (due to this identifying also the latest data file being  
20 sent to the web server). The subscriber's terminal sends a request for the video desired to the web server or the proxy. As a response, the terminal receives the index file that contains the information of the current video file (in addition, the invention can be used for any stream, not just video stream). When the subscriber's terminal knows the current video file, it can request  
25 the web browser to send, i.e. play, the first file. The subscriber's terminal can directly request the following files belonging to the video presentation. A presentation means a set of one or more streams presented to the client as a complete media show. The transport between the web server and the subscriber's terminal is made using the HTTP protocol. Furthermore when  
30 using a standard HTTP proxy the requests for media files are identified in a way that the proxy can store them, and only a single connection from the proxy to the web server needs to be made for multiple viewers using the proxy.

### Brief Description of the Drawings

In the following the invention is described in more detail by means of Figs 1 - 4 in the attached drawings where

- 5           FIG. 1    illustrates an example of the arrangement according to the invention,
- FIG. 2    illustrates an example of the data and index packets delivered from the broadcaster to the web server,
- FIG. 3    illustrates an example of a media stream send to a sub-
- 10          scriber, and
- FIG. 4    illustrates an example of a flow chart describing the method according to the invention.

### Detailed Description of the Invention

- 15          FIG. 1 shows an example of an arrangement according to the invention. Let a video provider be at a popular festival for making a live show for distributing the festival for subscribers. The cameraman takes shots, whose camera 1 is connected to a encoding element 2, called broadcaster. Broadcaster contains means 21 for defining video and audio devices, such
- 20          as the cameraman's camera, used to capture video information. The broadcaster supports different formats of video and audio, for example, DV or another format. It is also possible to define frame rate for the video to be captured. Audio and video are preferably synchronized.

- 25          The broadcaster encodes the captured video (and voice) using a encoding module 22. The video can, for example, be encoded to the MVQ (Motion Vector Quantization) format, but other suitable encoding methods can also be used. Since the encoding process is relatively tedious, and it is not convenient to send large video files via long transmission paths, the broadcaster is preferably situated near a place of shooting a video.

- 30          The broadcaster contains means 23 for sending the encoded video (and voice also) to a web server 3. The broadcaster uses regular FTP connection to send the streaming data (usually simultaneous video and voice streams) to the server, but another transmission protocol can also be possible. The sending means also contains connection options such as a destination address of the FTP server, i.e. the web server, bandwidth and a FTP
- 35          port. The user has to choose the name of the data stream, and a directory

with a name that is to be created on the web server. When first opening the FTP connection to the server, the broadcaster sends the web server an index file that describes the characteristics of the streams that are going to be sent over the FTP connection. The broadcaster sends the streaming data to the FTP server, i.e. the web server, which is identified by its IP addresses. The data are transmitted as soon as they are available (encoded and arranged into consecutive packets called fragments and segments) in the broadcaster.

There are actually two separate data streams for each transmission from the broadcaster to the web server as described in FIG. 2. The first data stream **221** contains small fragments (preferably about 2 seconds of data), and the second data stream **222** larger segments (preferably about 30 second of data). The lengths of the fragments and segments are configurable features. The small fragments allow the subscriber to start watching the video presentation without a considerable delay, while the larger segments reduce the load on the web server when the clients switch to download larger segments instead of the fragments.

One segment contains 15 fragments (in this case). The fragments are named so that the first fragment of the whole continuous stream is called 1-1.mvq, the second 1-2.mvq and so on, until all the fragments that fit into one segment have been sent. After the segment is full, the naming changes so that the change of the segment is indicated in the fragment name by increasing the first number of the name by one. So the first fragment in the second segment is named 2-1.mvq. Next one is 2-2.mvq and so on. The segments are also named sequentially so that the first segment in the continuous stream is called 1.mvq, the next 2.mvq, and so on. The naming of the fragments and segments makes it possible for the web server to deduce the current file and fragment in the segment.

The broadcaster starts to send both the fragments and the segments as soon as it has established an FTP connection with the web server and there exists valid encoded data to be sent. The Streams (consecutive transport packets containing data files) are sent to the server in the pre-chosen location and all the files are preferably saved into the same directory



As showed in FIG. 2, index files **223** (forming the third stream) are also sent to the web server. The index files are frequently sent at the beginning of each fragment file. The index file tells the web server the required information about the data files that the broadcaster is sending over the FTP connection. It should be noted that the index file does not need to transmit over a FTP connection, but it can be sent via another connection. The index file is named index.idx, and the first index file is initially sent to the server when the session, for example broadcasting, starts. The web server caches only the latest index file and is updated after either new fragment or segment is written to the server. The index file is overwritten each time a new fragment starts. For a video show request from the clients, the web server always reads the current data in the index file.

More specifically, the index file tells also what is the current segment and fragment number, and also the size (in bytes) of the fragments and the segments. Due to this the index file keep an order of the encoded data in the form of fragments and segments. Further, the index file tells how many fragments are included in each segment. This allows, for example, the player in the subscriber's terminal **41** to know when to switch from requesting the segments instead of the fragments. The web server then returns what it is asked. However, the web server uses index file data to know when the last fragment is finished, since it has to search for the next file to appear before it knows that previous file has been finished. Alternatively, each data file may end with known byte-sequence so that appearance of next file need not be known. As mentioned, the index file is constantly refreshed at the web server as new fragments and segments are written to the server.

The index file is located in the same directory wherein the data of the streams is stored. The index data has the following format:

```

Frag=num
Numfrag=num
Fragsize=num
Seg=num
Segsize=num
:
```

Frag specifies the current fragment number. Numfrag specifies the number of fragments in each segment. Fragsize specifies the size of the

fragment in bytes. Seg specifies the current segment number. Segsize specifies the size of the segment in bytes. The colon indicates the end of the index file. The colon is important, since index file is updated constantly, and web server might reach end-of-file before the new file had been fully written.

5           The web server 3 must be adapted to handle continuous media stream. The adaptation is achieved using a special CGI (Common Gateway Interface) program or server extension 33 (server extension such as Java Servlets, Apache modules, ASP) The CGI program accepts requests from subscribers for video show (streaming data form the broadcaster which is  
10 saved in consecutive files in the web server). It reads the data files as sent by the broadcaster, and transmits the files to the subscribers. The sending of the presentation files is performed such a way that the continuous presentation data, which is arranged into the file, is sent to the subscriber terminal in real-time before the whole presentation is formed. For achieving the current  
15 files CGI checks the status of the index file, which tells the current file. The information of the index is sent to the subscriber's terminal after which the terminal can directly request the current and following data files. The following data files can be deduced by the naming system of the fragments and segments. The index file is dynamically updated and it must be read each  
20 time when the CGI program is called.

          In other words, data stream is sent to the subscribers 4 in small fragments starting from the file specified by the index. Small fragment size reduces the starting delay. Every client starts at the beginning of the latest fragment, so delay at that point can be maximum of fragment size. Since  
25 there are other factors causing delay, such as encoding, transport and buffering, a few second delay caused by fragment size is acceptable) Later, larger segments are sent to reduce a number of HTTP requests. Using files to represent parts of broadcast allows requests for older data, and easy deletion of specified older parts of broadcasts. In addition, proxies can cache  
30 each part separately. Separate requests also make the system compatible with HTTP 1.0, since accessing parts of the files requires HTTP 1.1.

          In addition, the broadcaster generates an additional index file (FIG. 2, 224) for each segment to prepare video data for watching after live show. The segment index file is named x.idx. where x indicates the segment  
35 number. The segment index file tells the server the number of the segment

for which and the size of the segment in bytes. The segment index file has the following format:

```
Seg=num
Segsize=num
```

5

:

Seg specifies the number of the segment this index file is for. Segsize specifies the size of the segment in bytes. The colon indicates the end of the index file.

Data files are sent based on a request from a subscriber. The CGI program reads the file indicated by parameters, and reads the data file as it is written to the server directory. When reaching the first request for the getting a video representation the CGI program must read the index file send by the broadcaster. After this CGI gets direct requests for the latest video representation file. The CGI program accepts following parameters using HTTP GET method to allow caching: GET query parameters are not used. Instead, PATH\_INFO is used in the following format: `http://server/oplayo/live/video/videoname/seg/num/frag/num` Using query parameters would cause many proxies to think request to as dynamic preventing caching. Using path\_info turns parameters to look like directories and files, so proxies cache the response, and request only single copy of data for simultaneous connections through the same proxy. This increases the capacity of the system.

```
VIDEO=videoname
SEG=num
FRAG=num
```

25

VIDEO specifies the video to be sent. If no additional data is specified, the index data is sent to the subscriber. SEG specifies the current segment. This requires that the video be specified. The segment number is sent back to the subscriber. FRAG specifies the current fragment. This requires that the video and segment be specified. The fragment number of the segment is sent back to the subscriber.

Even though, using the CGI program means that a new process must be started for each request coming from the subscribers, the load and delay should be marginal. Further, since no server specific extension is used (only a server supporting a normal HTTP is required), the solution is not fixed to a single web server. Server extension can be used to improve perform-

35

ance. The important aspect of the invention is that data files are read as they are written in FTP upload directory, and that all data can be cached in proxies.

Instead of delivering a continuous data stream directly to a subscriber, the data stream can be delivered via a proxy 5 or a CDN (Content Delivery Network). A CDN is a service offered by a service provider. Fundamentally, a CDN maintains multiple locations with copies of the same content, and uses information about the user and the content requested to "route" the user to the most appropriate site. The proxy or CDN contains directory means 51 for saving media stream files.

FIG. 3 shows an example of a continuous media stream, which is delivered to a subscriber. Since the media stream is a live stream, a moment when the CGI program in the web server defines the current file to be delivered first is unknown. Let the moment represent fragment 7 311 in a segment. CGI delivers fragments 7 to 15 to the subscriber before it changes to deliver segments 312.

FIG. 4 illustrates the method according to the invention. First, material for a video (and voice for it) must be taken 411. When shooting the clips, the video and voice is captured by the broadcaster, which encodes the captured material 412. The encoded video and voice is arranged into consecutive packets 413, by forming an index file. The index file contains information of the current packet. The index file is updated frequently.

The consecutive data are streamed to the web server 414. This means that the connection between the broadcaster and the web server is established and the data are situated into transmission packets. The format of transmission packets depends of the transmission protocols used. Now the web server comprises the live stream of video show from where a subscriber may request it whenever desired.

When the subscriber requires the video show, CGI sends 415 the current index file as a response to the subscriber. After this the subscriber's terminal can directly request the current file from the web server where CGI defines the current file in the stream to be delivered to the subscriber.

In the subscriber's terminal 4, an applet or another means for requiring, receiving and playing video shows 41 downloads the data files based on the requests. The applet, called player, requires frequently the latest video show file. The first file is achieved using the index file. The follow-

ing files are achieved with direct requests concerning the files. The consecutive video files form a continuous data stream. Due to unexpected situations the player preferably buffers the data for a moment before playing.

As mentioned, 2 second files are downloaded before the following  
 5 30 second file starts. Longer files mean less requests and smoother functionality. Data files are asynchronously downloaded (independent of playing data), so that there are no stops. Small files are easier to handle in the cache than large files. And several subscribers who request the same video may actually get the same small file from the proxy. The CGI program follows the  
 10 frag number in the index file, and when it tells that the current segment is full, CGI knows that the next segment starts. This information can be used when the web server is switched to send segments instead of fragments.

Although above, there are described that the fragment and segment streams are needed for smooth function of the arrangement, alternatively only the fragment stream is required. The CGI program and the player  
 15 can be adapted to handle fragments as virtual segments. This can be achieved by utilizing the naming system of the fragments. After receiving the index file and first fragments the player can send requests only at the beginning of each virtual segment. The CGI program understands to send all  
 20 fragments belonging to the virtual segment as a response to the requests.

The invention makes it possible to use standard web servers that support an HTTP protocol. No separate streaming servers are required. No special components are needed in a proxy (or in CDN). Furthermore, the player is possible to switch to download small files, if the download speed is  
 25 too low.

In the broadcaster, it is also possible to save live streams and play them at a later time. The name of the save file must be specified before starting the broadcast. The saving function may be turned off or on during the broadcast transmission or the saving can be automatic. It may also be possible  
 30 to construct a stream that contains only video or audio. An audio codec can be selected. Furthermore, the video input can be filtered before encoding. The filter reduces the noise that is in the picture information and thus smoothes the picture.

Since the invention utilizes a standard HTTP protocol, firewalls  
 35 (FIG. 1, 6) are not a problem for receiving a live video show. Most firewalls probably let HTTP messages go through. In addition, many firewalls have



5

10

[illegible]

### Claims

1. An arrangement for delivering at least one live presentation to at least one subscriber via a network that comprises at least one web server, characterized in that the web server comprises a delivery module  
 5 which receives continuous presentation data from a encoding element, arranges the continuous presentation data into presentation files, defines a current presentation file, handles requests of the subscriber for getting the presentation, and delivers the presentation files as a response of the requests starting from the current file, the encoding element forming independent functions of receiving the continuous presentation data and sending the  
 10 presentation files.

2. An arrangement according to claim 1, characterized in that the sending of the presentation files is performed such a way that the continuous presentation data which is arranged into the file is sent to the  
 15 subscriber terminal in real-time before the whole presentation is formed.

3. An arrangement according to claim 2, characterized in that an index, which is updated frequently, containing information of the current presentation file is used for defining the first presentation file for the delivering to the subscriber.

20 4. An arrangement according to claim 3, characterized in that the index is an index file.

5. An arrangement according to claim 4, characterized in that the index file is created, and sent frequently to the web server in the encoding element.

25 6. An arrangement according to claim 5, characterized in that the request is an HTTP GET method containing path info for the presentation.

7. An arrangement according to claim 6, characterized in that the path info contains the name of the presentation.

30 8. An arrangement according to claim 7, characterized in that after receiving the request the delivery module sends the index file as a response to the subscriber.

9. An arrangement according to claim 8, characterized in that after receiving the index file a player module in the subscriber terminal requests the first file of the presentation, the request containing the name of  
 35 the presentation and an identification path information of the current file.

10. An arrangement according to claim 9, characterized in that following presentation files after the first file is deduced by the player module, the player module requesting the following files from the web server. the request path info containing the name of the presentation and identification path information of the latest file for the presentation.

11. An arrangement according to claim 10, characterized in that the identification path information contains fragment and segment information of the presentation file, the segment information indicating a part of the continuous data and the fragment information indicating a part of the segment information.

12. An arrangement according to claim 11, characterized in that if the identification path information fails to contain fragment information, the presentation files belonging to the segment is sent as a response to the player module.

13. An arrangement according to claim 2 or 12, characterized in that the delivery module contains a directory module for saving the presentation files.

14. An arrangement according to claim 2, 12 or 13, characterized in that the delivery module is a CGI program.

15. An arrangement according to claim 2, 12 or 13, characterized in that the delivery module is a server extension.

16. An arrangement according to claim 5, characterized in that the encoding module contains means for creating the index file and arranging the continuous data into entities defined in the index file.

17. An arrangement according to any of claims 11-15, characterized in that the player module contains means for switching to request segments instead of fragments.

18. An arrangement according to any of claims 1-17, characterized in that the arrangement comprises a proxy through which the presentation files are delivered from the web server to the subscriber terminal, the proxy being capable to save the presentation files.

19. A method for delivering a live presentation to at least one subscriber via a network, characterized in that the method comprises the steps of:

a) receiving a continuous media stream and identifying information of a current presentation data in real-time in a web server



b) forming presentation files from the continuous media stream and saving the real-time identifying information, and

c) delivering the presentation files as a response to a request for the live presentation in such a way that the identifying information tells the first file to be delivered, and after this following requests identify directly the following presentation files to be delivered.

5

20. A method according to claim 19, characterized in that the delivering step of the presentation files is performed such a way that the continuous presentation data which is arranged into the file is sent to the subscriber terminal in real-time before the whole presentation is formed.

10

21. A method according to claim 20, characterized in that the prior step a) the method comprises the steps of:

forming in real-time a continuous media stream, and

identifying frequently a current presentation data in the media stream.

15

22. A method according to claim 20, characterized in that the requests are HTTP request comprising path info of the presentation files.

23. A method according to claim 22, characterized in that the depending on the path info of the request a single presentation file or a group of presentation files are delivered as a response to the request.

20

24. A method according to claim 20 or 23, characterized in that the delivering the presentation files is made through a proxy, which is capable to save the presentation files.

25. A method according to claim 24, characterized in that the proxy multicasts the live presentation utilizing the saved files.

25

30

(Fig. 1)

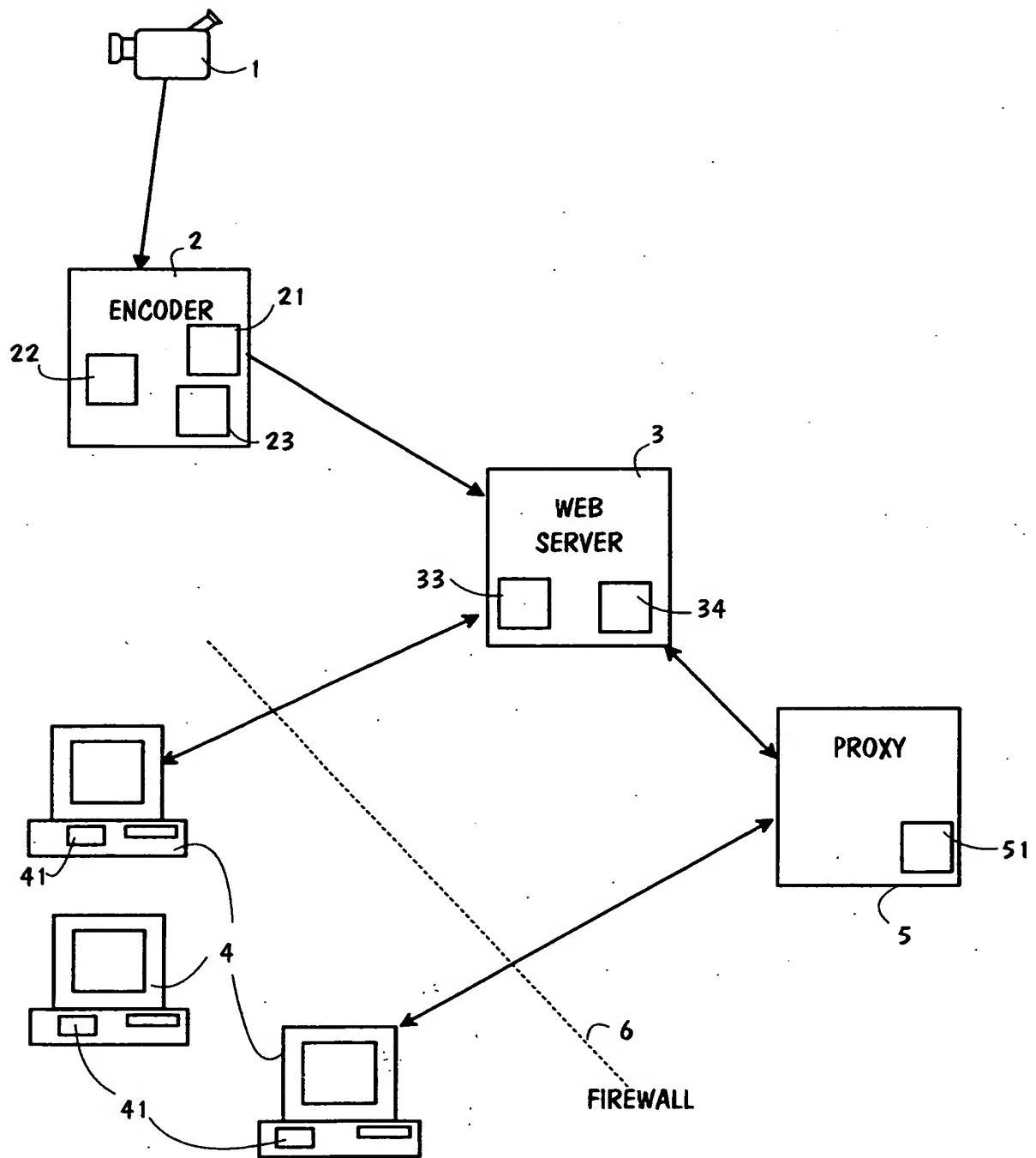


FIG. 1

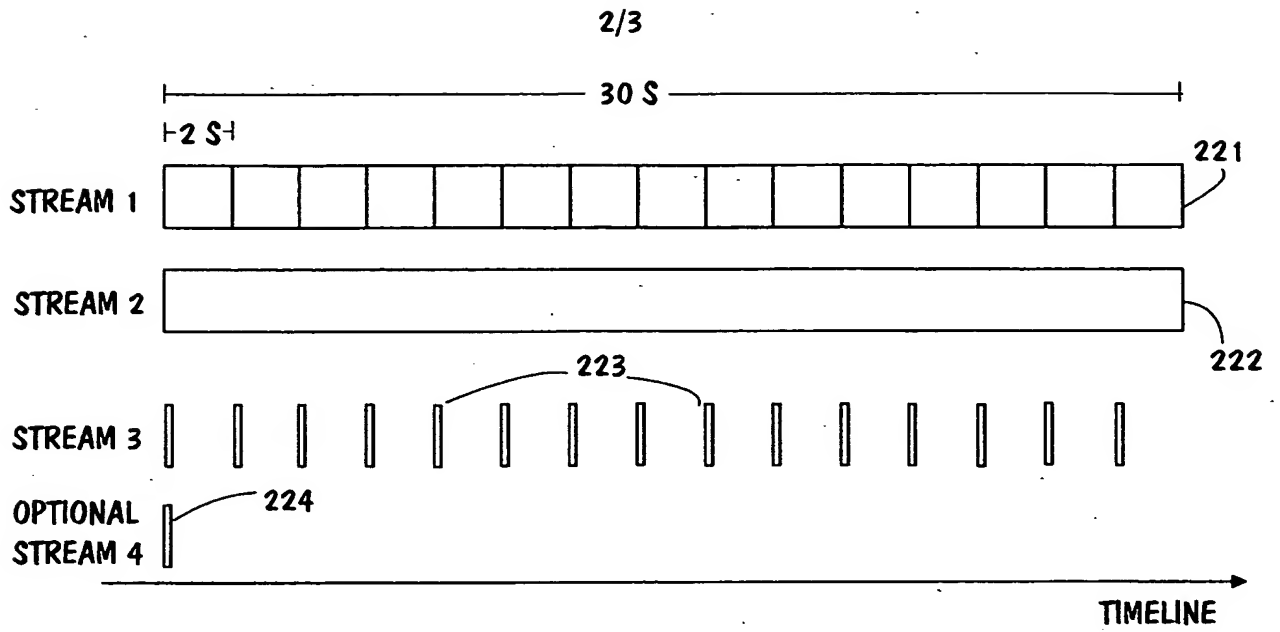


FIG. 2

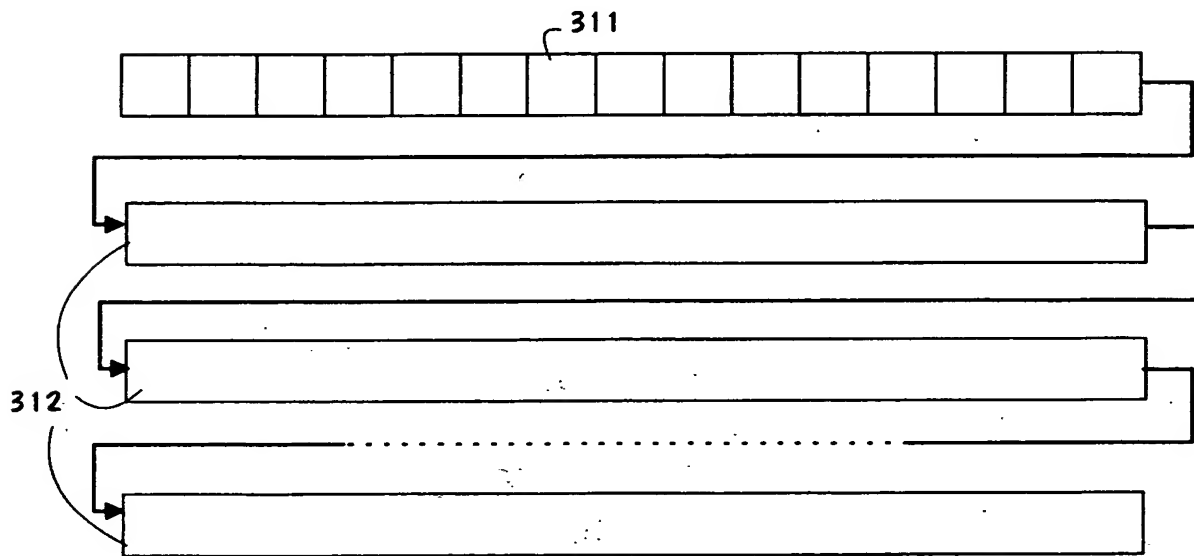


FIG. 3

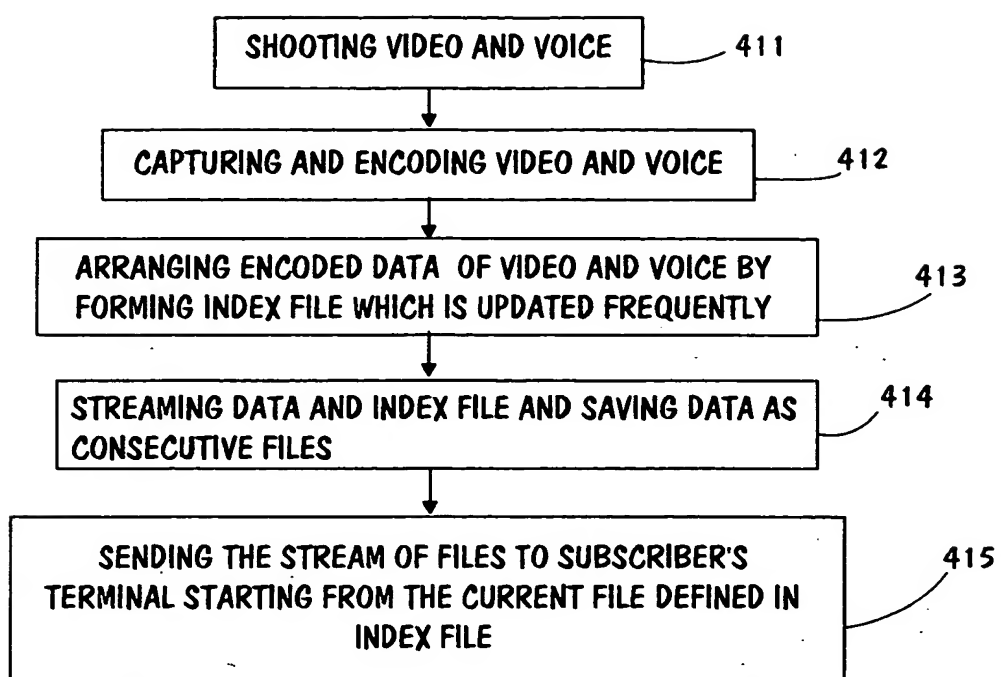


FIG. 4